AST 303 - Fall 2016

# Some useful information about using the Peyton Hall computers

Welcome to AST 303. These notes should help you get started with computing in Peyton Hall and the `python` programming language.

You should all have an account on the Peyton computing system. If you don't, please let us know. Extensive documentation about using the Peyton network is available at
`http://www.astro.princeton.edu/docs/`

You can access these computers in two ways. In Peyton Hall, you may use any unoccupied terminal in Rooms 29 and 30 (in the basement). Remember to log out when you are done, so that others can use the terminal! Alternatively, you may log onto one of the computers from outside Peyton Hall via 'SSH', a protocol that allows you to connect to computers over the internet. Setting up SSH is described at: `http://www.astro.princeton.edu/docs/SSH`; the best computers to log onto from outside the building are trek.astro.princeton.edu and raleigh.astro.princeton.edu (or any of the machines in the undergraduate cluster).

If you are unfamiliar with Unix (the operating system used by the Peyton Hall computers), please look at the various resources linked from the course home page which describe it. Unix is very similar to the OS X operating system used by Mac computers. One thing you will need to learn is a way to create and edit files; emacs is the recommended choice (simply type
`emacs ⟨filename⟩`
at the command line). If you simply type `emacs` (without a filename), a window will open that will point you to a pedagogical tutorial to get you started.

While `python` (the computer code we will use extensively in this course) is broadly available, the version we have here on the Peyton Hall computers includes all the extensions you will need (including `numpy`, `scipy`, and `matplotlib`). You may need to install those extensions yourself if you decide to work on your own machine.

There are two ways to get python started:
**Method 1:** In order to use python on the computers in Peyton Hall, you must first type the following line:

```
module load python
```

This invokes python version 2.7.8.

You can put this line in your .bashrc or .cshrc file, so it is invoked automatically every time you log in.

**Method 2:** Use the so-called anaconda python distribution, as follows:
On both the Peyton Hall computers, and your laptop, download the installer shell script from
`https://www.continuum.io/downloads`
(get the python 2.7 version). Once you've done this, you can use `python` (now living in anaconda/bin/python in your home directory).

When using `python` interactively, we recommend that you use `ipython` (see anaconda/bin/ipython), instead of the standard `python`; it has a number of features that make it much easier to work with interactively. Now you can start familiarizing yourself with the language, by typing in some arithmetic operations and see that you get the right results.

An excellent tutorial on scientific use of `python` is available at:
`http://scipy-lectures.github.com/`
If you have never used `python` before, we recommend you go through the first few chapters to get familiar with the basics. There are other tutorials and guides to `python` linked from the course home page. Appendix A of the Ivezić et al. textbook also is enough to get you started, and the recommended textbook by Kinder and Nelson gives you much more.

The above-mentioned `python` extensions, `numpy, scipy` and `matplotlib` are also covered in the above tutorial. In particular, `numpy` and `scipy` commands are optimized to work with arrays of data, so you should always prefer these to the standard `python` commands whenever possible (e.g. use `numpy` arrays instead of `python` lists for vectors/matrices containing data! With `numpy` arrays, a command adding together, for example, the elements of two arrays of 100,000 elements will be done as a single operation, rather than involving a loop over all 100,000 elements.) The `matplotlib` library allows one to use `python` to make plots. A nice introduction to it is in Chapter 1.5 of the tutorial mentioned above. Another good resource is
`http://matplotlib.org/users/pyplot_tutorial.html`

To use these extensions, you will want to put the following at the beginning of all your `python` scripts:

```
import numpy as np
import matplotlib.pyplot as plt
```

This imports all the commands of these extensions, and allows you to reference them without typing `matplotlib.pyplot` every time you need to use one of them[1].

In order to print a copy of a figure that you have created, you will first need to save it as a file. You can do this with the `savefig` command:

```
plt.savefig('figure_name.pdf')
```

which creates a .pdf file in the directory in which you're working with the name `figure_name.pdf` . To send this figure to a printer in Peyton, plot it (outside of `python`) with the `acroread` command:

```
acroread figure_name.pdf
```

(this brings up a copy of the figure) and click on the print option within `acroread`. A list of printers in the department can be found in the Peyton documentation at
`http://www.astro.princeton.edu/docs/Printing`. There is one, called 'hp576', for example, in Room 30.

There are links to many other useful python resources on the course website.

[1]`scipy` is huge, and so rather than importing the whole thing, you will import specific routines from `scipy` when you need them.